DADA: A 2-Dimensional Adaptive Node Schedule to Provide Smooth Sensor Network Services against Random Failures

Shengchao Yu, Antony Yang, and Yanyong Zhang

Department of Electrical & Computer Engineering
Rutgers, The State University of New Jersey
Piscataway, NJ 08854
{yusc,pheroth,yyzhang}@ece.rutgers.edu

Abstract

It has been recognized that sensor network deployment should include a high degree of redundancy so that every node can be covered by a few others and they can take turns to perform the duty. This strategy can potentially help extend network lifetime and maintain a desired coverage and connectivity level. In order to achieve this goal, one must carefully schedule the actions of the redundant nodes to simultaneously address energy conservation, coverage, and connectivity. This paper presents the design and implementation of a 2-dimensional adaptive technique (DADA) that can dynamically adapt a redundant node's schedule based on application demands and network conditions. DADA introduces both spatial diversity and temporal diversity to the node schedules, thus leading to a much prolonged lifetime. At the same time, it provides protection against unexpected node failures. Using extensive simulation studies, we demonstrate that our schemes can improve the network lifetime by a factor of 2.9 compared to ASCENT, while providing improved network coverage and connectivity by better surviving unexpected node failures. Further, we demonstrate that the lifetime improvement of our schemes scales linearly with the deployment redundancy.

1 Introduction

Recent advances in MEMS technology, and wireless communication and networking, have enabled the development of relatively low-cost and capable wireless microsensors, thus bringing on a new class of applications involving event surveillance and date collection [6, 13, 9]. However, to deploy these remote sensing applications in large scale, there are still barriers to overcome. One significant barrier essentially boils down to the fact that the lifetime of a sensor network is still too limited to function over a significant period of time. Extending network lifetime is challenging because these networks are built out of very short-lived sensor nodes. The lifetime of a sensor node is limited by its battery capacity, and, due to limitations of cost and size, the sensor hardware is rather unreliable. Additionally, sensor networks are often deployed in harsh environments and left unattended after deployment, which further contributes to frequent node failures.

Over the past few years, a considerable amount of research effort have been devoted to address this challenge, which has fostered a family of heuristic-based strategies and analysis [2, 27, 26]. These strategies share the viewpoint that sensor network deployment should include a great degree of redundancy, so that every node is covered by a few others and they can "take turns" to perform the duty. Although these studies have revealed valuable insights to sensor network deployment and planning, network lifetime remains a hurdle for it is still unclear how the redundant nodes should take turns to minimize the energy spent in maintaining a desired coverage and connectivity level.

One solution that has been proposed for surveillance applications using sensor network [26] is to adopt a fixed round-robin-like schedule between a group of redundant nodes, which specifies when a node should become active to work and how long it should stay active. A fixed schedule suits scenarios with low network dynamics, but it is not sufficient to handle scenarios with high dynamics, like network topology variation due to battery depletion and hardware failure, fluctuating channel quality, and traffic volume shift.

As a response, several distributed algorithms have been proposed to extend lifetime for highly dynamic networks, which are referred to as *distributed wakeup* in this paper. In this type of algorithms, nodes can independently sleep for some time, and then wake up to check whether it needs to participate in monitoring or routing at the moment. If not, it goes back to sleep again. Distributed wakeup strategies can better tolerate network dynamics compared to fixed round-robin schedules, without assuming the status of other nodes. However, one must carefully tune the duration of a node's sleep interval for a long sleep interval may lead to loss of coverage or connectivity while a short sleep interval may nullify the benefit of redundant deployment. Most existing distributed wakeup schemes try to make the sleep intervals uniformly small for all the sensor nodes in order to make the network responsive to unexpected changes. Obliviously, such a simplistic scheme involves frequent wake-ups and status checking and can lead to excessive energy consumption.

Instead of a context-oblivious wakeup method, we believe a node should dynamically adapt its sleep interval based on the application demands and the network conditions. As a result, sleep intervals seen in the network exhibit diversities both spatially and temporally: spatial diversity meaning each node may have a distinct sleep interval based on parameters such as their energy level, functionality, neighborhood density, and spatial proximity to the event spot or routing paths; and temporal diversity meaning a node will adopt different sleep intervals at different points in its lifetime.

Inspired by the basic idea of two-dimensional adaptivity (*DADA*), we propose two heuristics, *satellite* and *asynchronous wake-up*. In satellite technique, one or more satellites are elected

to cover every active node, and they will stay alert by waking up frequently. Whenever necessary, these satellites can become active and join the network activity at little cost. As a result, the rest of the redundant nodes can sleep for a much longer period, leading to a substantial energy conservation. On the other hand, asynchronous wake-up lets every redundant node make an independent decision about its sleep interval based on its local perception of the network conditions.

In this paper, we present the basic framework and considerations that are involved in deploying these two heuristics. We also discuss the detailed evaluation results using ns-2 simulator. We compare the two techniques of DADA with existing strategies such as the one proposed in [2], and we have found out that our schemes can improve the network lifetime by a factor of 2.9, while still providing a better network coverage and connectivity. More importantly, we have shown that our schemes result in an almost linear improvement in network lifetime as the redundancy degree increases, while existing schemes cannot improve lifetime after the redundancy degree reaches certain level.

The rest of the paper is organized as follows. Section 2 describes the specific system models and assumptions we have made in our study. A brief summary of related work in conserving energy in wireless ad-hoc network (including sensor network) is presented in Section 6. The details of the two proposed schemes are discussed in Section 4. The detailed evaluation results are summarized in Section 5. Finally, Section 7 gives the concluding marks and future direction.

2 Problem Setup

Sensor networks can find a wide range of applications, such as habitat monitoring, fire detection, at-risk heart supervision, and highway traffic surveillance, to name just a few. Although these applications significantly differ from each other, they all impose one fundamental requirement to the underlying networks, namely, the continuous provisioning of both coverage and connectivity.

It is hard to adopt a uniform coverage model for the entire spectrum of sensor applications, as different application characteristics and network configurations may have different coverage requirements. For example, some applications only require to monitor a few interesting spots in the field, while others may require to cover every inch of the network field. However, a generic framework that can be used to capture the coverage requirements of a significant number of applications, referred to as *grid-based* coverage model, has been proposed by He in [26], illustrated in Figure 1. In this framework, the entire sensor field is represented by a virtual grid, and each grid point must be covered by at least one active sensor node. In Figure 1, a circle depicts the sensing range of the sensor node that is located at its center (with radius r). By varying the grid size, we can model diverse coverage requirements, ranging from a coarse-granularity coverage (corresponding to a large grid size) to a fine-granularity coverage (corresponding to a small grid size). In this paper, we use this coverage model to help explain the design details of DADA, and further implement this coverage model in our simulator to evaluate DADA's performance. DADA, however, is not limited to this model, but it can work with other coverage models (e.g., [27]) as well with minor modifications.

Under certain circumstances, these applications require the sensed data be delivered back to the data sink. For instance, a fire-detection application in a forest may require its temperature sensors to send back their readings when the readings are above 100 degree. As a result, some of the active nodes that monitor the network field will become sources and start reporting its data to the corresponding sink(s). Once the source nodes start sending data, a continuous network connectivity must be guaranteed. Again, we assume that a sensor node's radio/communication range is a circle with radius R, centered at itself.

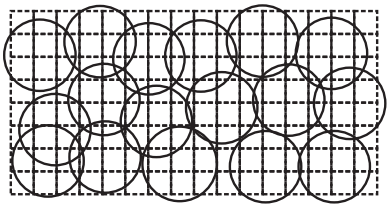


Figure 1: The grid-based coverage model. The sensor field is represented by a virtual grid, and each grid point must be covered by at least one active sensor node. A circle represents the sensing range of the sensor node located at its center.

Sensor network deployments usually employ a large degree of redundancy. However, it is unnecessary for all the sensor nodes to stay active and participate in network operations simultaneously. Instead, at any instant, a sensor network should only have a subset of active nodes, while others, referred to as redundant nodes, can stay asleep (with their radio off) to conserve energy and extend network lifetime. An active node, either monitoring the field or participating in routing, can be replaced by, and thus associated with, one or more redundant nodes. In our grid-based coverage model, suppose active node i covers grid points $G_i = \{i_1, i_2, ..., i_n\}$, where n is the number of grid points i covers. Therefore, any redundant node that can cover a subset of G_i is considered to belong to i's coverage redundant node set. Similarly, all the redundant nodes that are located within the radio/communication range of i form i's connectivity redundant node set. On the other hand, from the viewpoint of a redundant node, it can belong to several active nodes' redundant node sets.

In order to maximize network lifetime, we usually keep a minimum set of active nodes within the network ([22, 10, 4, 11, 17, 23]). Consequently, if any of the active nodes fails (either due to energy depletion or due to unexpected failures), the network may experience a temporary loss of coverage or connectivity. In order to ensure a smooth recovery, as soon as an active node fails, we should make one or more its redundant nodes awake so that the network operations can resume quickly. Since we may need more than one node to completely cover a sensor node's sensing range or communication range, we define k as the recovery degree, and we try to wake up k redundant nodes when an active node fails.

DADA's goal is to provide a continuous coverage/connectivity for sensor networks, regardless of how often active nodes fail, while extending their lifetimes. Specifically, it addresses the following two challenges:

- *To make k redundant nodes awake after a failure.* A redundant node turns its radio off when in sleeping, and it is thus impossible to "wake up" a sleeping node. DADA proposes two heuristics that try to make sure we have k awake redun-

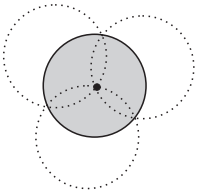


Figure 2: If the current active sensing node fails, we may need at least three nodes to maintain the same coverage.

dant nodes when active nodes fail. At the same time, both heuristics also try to minimize the periods of time when the redundant nodes stay awake before their active nodes die because a considerable amount of energy will be consumed if the radio is on.

- *To determine when redundant nodes need to become active.* If a redundant node stays awake all the time, and keeps listening to the channel, then it knows when it should become active on a real-time base. However, this method can lead to an excessive energy consumption. Instead, redundant nodes should only periodically check whether they should become active or not. DADA proposes an effective framework for this purpose.

In the following two sections, we discuss how DADA addresses these two challenges respectively.

3 Contributions of DADA: Coverage, Connectivity, and Lifetime

In this section, we discuss how DADA can help maintain coverage, maintain connectivity, and extend lifetime.

3.1 Coverage

The necessity of extending network lifetime demands a minimum set of nodes that stay active and perform either sensing or forwarding duties. Once an active sensing node dies, we must wake up one or more redundant nodes to cover its sensing area (a circle with radius r centered at the node itself), such that network coverage can be maintained. In [4], Gao has shown that sensing range of any node can be covered by 3-5 nodes that are located within its sensing range. Figure 2 illustrates this idea. Hence, whenever an active node i dies, it is imperative for DADA to make sure that: (1) at least 3-5 redundant nodes that are located within i's sensing range are made awake, which become i's coverage backup set, and (2) a minimum set of nodes are chosen from the backup set to maintain coverage and others can stay in energy-saving mode.

Selecting proper nodes from the backup set is beyond the scope of this paper. In fact, there has been a rich literature body [22, 10, 4, 11, 17, 23] that tries to address this problem. The goal of DADA is to make enough nodes available quickly in an energy efficient way so that the "coverage calculation" unit can easily select appropriate ones to continue the sensing task. After the selection is performed, the extra active nodes can go back to sleep.

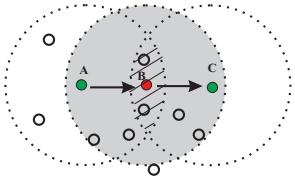


Figure 3: Connectivity maintenance

Now let us look at DADA's framework on how to make enough backup nodes awake upon failure. Every time when a redundant node wakes up, it must inquire if there is an active sensing node within its own sensing range. It needs to temporally adjust its transmission power so that it can broadcast a message within the radius r (instead of its radio range R). Most of the sensor radios can adjust their transmission power. For instance, an 802.11 card can do so by making an `ioctl` call, and a Berkeley mote can change the transmission power by the statement `CC1000Control.setRFPower(char value)`. After it sends out the query message, if it does not receive any response within a time period, it concludes that there is no active sensing node within its range, and it will become awake.

After a redundant node just wakes up, it will keep silent for a period T_{silent}, not responding to any subsequent query messages from other inquiring redundant nodes. The silence period ensures that more than one redundant node can be made awake. The set of redundant nodes that are awake are referred to as the *coverage backup set*.

In section 4, we present two techniques that enable timely formation of coverage backup set upon failures, at a low energy budget.

3.2 Connectivity

Let us use the example illustrated in Figure 3 to explain how DADA can maintain network connectivity in an energy-efficient manner. In this example, we had a routing path $\overrightarrow{ABC}$ before node B dies. As soon as B dies, DADA (1) wakes up one or more nodes which are within either A's or C' radio range; and (2) selects appropriate ones from the awake nodes to maintain connectivity.

Unlike in coverage maintenance where redundant nodes find out whether they need to become active in a proactive fashion, here, redundant nodes rely on the active routing nodes to notify them. Specifically, in this example, A and C need to detect B's failure and try to wake up the redundant nodes. Between nodes A and C, we claim that it is more difficult for C to detect B's failure. First of all, many sensor applications do not require regular periodic data reporting. Instead, they go through periods with high traffic volume and periods with low traffic volume, even sometimes periods during which there might be no traffic. As a result, it is not straightforward for C to differentiate B's failure to forward due to failure from a pause in source reporting. In order to address this problem, [2] has proposed a detection scheme based on sequence numbers. However, we argue that if B dies, C may not hear anything, so sequence numbers do not help as well. Another possible way for C to detect is to probe B after it has not heard from B for some time. This scheme ensures detection, but the detection may not be timely because probing can only start after some delay. On the other hand, A can easily find

out whether B is functioning or not by observing the acknowledgments from B employed by the MAC layer protocol (it has been pointed out in [21, 28] that acknowledgments are necessary for reliable delivery even in sensor networks).

As soon as A detects B's failure, A periodically broadcasts "help" messages to its vicinity. The redundant nodes that happen to receive the help messages when they are awake will join the *routing backup set*. From the nodes in this backup set, the routing protocol can build a new route quickly. After the new route is established, the unused nodes can go back to sleep. We would like to emphasize that making multiple redundant nodes awake is critical to connectivity provisioning because it may not be possible to build a valid route from a single node, even if it is within B's radio range. Referring to Figure 3, it is obvious that only those nodes that are within the intersection of radio ranges of A and C can continue the routing alone; for nodes in other region (still within B's radio range), we need more than one node to build a new path.

Again, DADA can borrow ideas from existing routing techniques to build a valid route. Much of the earlier work has investigated energy-efficient routing alternatives, such as Directed Diffusion [8] and RAP [12], to name just a few.

3.3 Lifetime

A sensor node can operate in various states. According to different levels of energy consumption, we identify the following four important states: (1) transmitting a message, (2) receiving a message, (3) idling/listening, and (4) sleeping with radio off, with p^{xmit}, p^{recv}, and p^{idle} being the power level required by a node in states (1), (2) and (3) respectively. A sensor node in state (4) has much less power consumption p^{sleep}, usually around 1% of p^{xmit}, as shown in [14, 18]. Since p^{sleep} is negligible, we consider p^{sleep} as 0 in the rest of analysis.

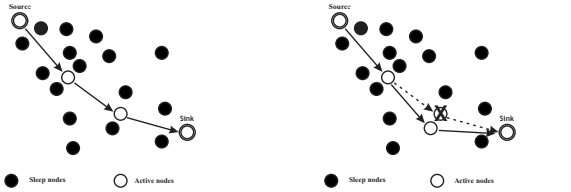
A sensor node has an initial energy E (in Joules). In this study, we employ the linear battery model in which the battery is treated as linear bucket of energy [16]. If a sensor node spends times t^{xmit}, t^{recv}, t^{idle} and t^{sleep} in states (1), (2), (3) and (4) respectively during its lifetime (T), then $T = t^{idle} + t^{recv} + t^{xmit} + t^{sleep}$, and $E = t^{xmit} \times p^{xmit} + t^{recv} \times p^{recv} + t^{idle} \times p^{idle}$.

Lifetime of a sensor network is governed by the average energy consumption rate of the network. The average energy consumption rate (r) is calculated as

$$r = p_a \times r_a + (1 - p_a) \times r_r,$$

wherein p_a denotes the average percentage of nodes that stay active at any time, r_a the average energy consumption rate for an active node, and r_r the average energy consumption rate for a redundant node. Usually we have $r_a > r_r$. Among the three factors, r_a is decided solely by application behaviors such as required data reporting rate and routing protocol. Instead, DADA can help reduce both p_a and r_r, leading to a longer lifetime:

- DADA reduces p_a. Whenever an active node dies, DADA can wake up more than one redundant node, from which we can choose one to minimize p_a.
- DADA reduces r_a. A redundant node goes through cycles of sleeping, waking up, and listening before it becomes active. DADA can make redundant nodes wake up less often when they are not needed, thus resulting in a smaller r_a.



(a) At time t_0, only four nodes are active (including the source and sink).

(b) At time $t_0 + \delta$, one of the active nodes dies. At the same time, a nearby sleeping node wakes up and joins the routing.

Figure 4: An ideal sleep interval.

4 Adaptive Node Scheduling Techniques

DADA has two over-arching goals: quick network recovery from random node failures, and prolonged network lifetime. In order to achieve these two goals, our viewpoint is that we should let redundant nodes sleep as much as possible when they are not needed, while wake them up immediately upon node failures. There are different ways of implementing this viewpoint, and we propose two algorithms: satellite algorithm and asynchronous wakeup algorithm.

4.1 Basic Idea

first design issue is after a node becomes active, whether it should stay on duty until it dies [27, 2], or several nodes should take turns to perform the duty by time-slicing [26]. DADA takes the former option because it incurs less configuration overhead due to the change of participating active nodes. As an example of configurational overhead, many routing protocols need to be re-configured if the involved nodes change. These overheads can lead to periods that are not responsive.

A redundant node alternates between sleeping periods and awake periods. When it is awake, it checks whether it should become active (Section 3). An active node will stay active until it fails. A node's sleep interval governs how often it wakes up, and it is thus the most important parameter in balancing network responsiveness and network lifetime. Figure 4 illustrates an ideal sleep interval for maintaining connectivity. We use this example to illustrate the philosophy for our waking up heuristics: a node should sleep shorter when it will be needed soon, and sleep longer otherwise. The basic idea of DADA is that every awake redundant node estimates the likelihood that it will be needed in the near future, and then adapt its sleep interval accordingly. As a result, the entire network should not adopt a fixed sleep interval, nor a single distribution. Instead, redundant nodes should adapt their sleep intervals both spatially and temporally. We first briefly summarize the two proposed heuristics with respect to how they achieve adaptivity in these two dimensions, and then we present the details of the algorithms in the following subsections:

- *Spatial Adaptivity - sleep intervals exhibit spatial diversity.*

  The two proposed heuristics use different approaches to achieving spatial diversity. In satellite scheme, every active node selects a small number of "satellites" from the redundant nodes which will stay alert by frequently waking up.

Consequently, the rest of redundant nodes can sleep more aggressively, and they wake up in a synchronous fashion. Essentially, this heuristic partitions the entire set of redundant nodes into small groups so that nodes within the same group can cover each other and the active node(s), thus allowing other groups to conserve more energy.

On the other hand, asynchronous wake-up scheme takes the viewpoint that every redundant node should make an independent decision based on its local conditions such as how many redundant nodes an active one has.

- *Temporal Adaptivity - the same node may have different sleep intervals at different points in its lifetime.*

  During a node's lifetime, it should adapt its sleep intervals based on the network configurations at different instants. For example, if the nearby active node's remaining energy is lower than a threshold, it should wake up more often. By employing temporal adaptivity, we can have more than one awake redundant node when the active node dies, so that the network can quickly resume its normal function.

4.2 Satellite Algorithm

One intuitive way of providing smooth network coverage/connecitivy is to have all the redundant nodes stay awake, so that they can join active nodes whenever they are needed. The obvious downside of this approach, however, is that the network lifetime will be significantly shortened because of the excessive energy consumption for keeping all the nodes awake. To balance these two aspects, instead, we can keep a subset of redundant nodes awake, and the rest of the redundant nodes can sleep for a much longer period. Those redundant nodes that stay awake act like "satellites" of the active nodes, and this algorithm is thus referred to as *satellite* algorithm in this paper.

Satellite Election First, let us look at how we select satellites from the redundant nodes. For the sake of simplicity, let us consider one active node. Specifically, that active node has n redundant nodes, from which we would like to choose k satellites. At the beginning, we assume all n redundant nodes are awake, and each redundant node needs to register itself with the active node. There are a wide range of criteria one can use to choose satellites, different criteria suitable for different application requirements and network configurations. To name just a few, these criteria include: (1) the remaining energy of the satellite should be greater than or comparable to that of the active node, so as to ensure that satellites survive the active nodes; (2) satellites should be able to replace the active node in terms of coverage and connectivity capability; and (3) satellites should not be too spatially close to each other such that they will not fail at the same time due to attacks (either natural physical phenomena or attacks by malicious adversaries). Some criteria are easier to examine than others; for example, in order to examine (3), each node needs to know its location. Some criteria even conflict with others, e.g., (2) and (3) in the above list. The point we would like to make here is that it is not DADA's job to come up with these criteria, but the applications should adopt appropriate ones considering its characteristics and what is available in the network (e.g., location awareness). After the active node chooses k satellites, it informs them, and put the rest of the redundant nodes into sleep. The satellites are there to guard against random node failures. If an active node fails, its satellites can quickly detect the failure and resume its functions. The redundant nodes that are not satellites sleep for a long period of time, and then wake up at the same time.

Lifetime Estimation Next, we look at how long the redundant nodes should sleep. Our viewpoint is that since the active node is guarded against by its satellites, then the redundant nodes can sleep for the remaining lifetime of the active node, which requires us to estimate a node's remaining lifetime. Our estimation method is based on the observation that physical phenomenon is usually a continuous function, so that the sensing rates and communication rates are continuous as well. As a result, we use the average communication rates over the past to predict the future communication rates. We would like to emphasize that our estimation method is intended to provide a reference point to the redundant nodes' sleep time, so its accuracy is not required.

Suppose a node has been active for time T and it has sent N messages during that period. Further suppose that it has sent n messages during last t time units. Then the estimated send rate in the next T' period is calculated as

$$s' = (1-\alpha) \times \frac{N-n}{T-t} + \alpha \times \frac{n}{t}, \tag{1}$$

where α is a parameter which notifies the significance of the recent history to the future estimation.

Suppose a node has remaining energy E', and its remaining lifetime is T'. If that node spends all the remaining time idling, then the total energy consumption is:

$$E^{idle} = p^{idle} \times T'. \tag{2}$$

At the same time, Equation 1 gives the estimated send rate of that node. Since packet sizes in sensor networks are usually small and fixed-sized [15], the corresponding energy consumption for sending can be calculated as:

$$E^{xmit} = p^{xmit} \times s' \times T' \times t_0, \tag{3}$$

where t_0 is the time involved in transmitting a packet. We should reiterate that this study's primary focus is to design energy conservation protocols when the traffic volume is relatively low. Equation 3 does not consider energy involved in retransmitting collided packets for this reason.

When traffic is relatively low, the incoming traffic rate is equal to the outgoing traffic rate. Consequently, we can calculate the energy consumption in receiving mode as:

$$E^{recv} = p^{recv} \times s' \times T' \times t_1, \tag{4}$$

where t_1 is the time to receive a packet.

Since the node will spend a large fraction of its lifetime idling (low traffic volume assumption), its total energy expenditure is:

$$E = E^{idle} + E^{xmit} + E^{recv}. \tag{5}$$

As a result, the remaining lifetime of the routing node is:

$$T' = \frac{E}{p^{idle} + s' \times t_0 \times p^{xmit} + s' \times t_1 \times p^{recv}}. \tag{6}$$

Finally, the active node can set the sleep time as $k_1 T'(0 < k_1 < 1)$ where k_1 is a tunable parameter.

As described above, the current active node needs to collect statistics such as T, N, t, and n to estimate its remaining lifetime. If it just became active, it should inherit these statistics from the previous active node (for which it was a satellite). Hence, the active node and its satellites should synchronize with each other from time to time to share these states.

Integration for Multiple Active Nodes In the above discussion, we assume that there is only one active node. Next, we discuss how to integrate the schedules for all the active nodes.

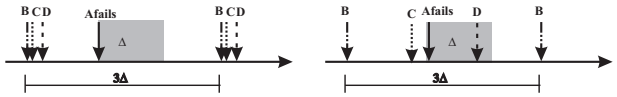
When we move from one active node scenario to multiple nodes, we need to address two issues. The first issue is whether two active nodes should share the same satellites or they should have a disjoint set of satellites. As an example, let us consider the following scenario: A and B are two active nodes, and C is a common redundant node for both of them. C became A's satellites at time t_A. At time t_B ($t_B > t_A$), all of B's redundant nodes wake up and B needs to choose a new set of satellites. If B shares C with A, it can conserve more energy, but the disadvantage is that it can only provide a weak guarantee on network recovery. Once C becomes active for A, B will lose one of its satellites. Hence, in our algorithm, each active node should have its own k satellites. Although each active node must have its k satellites, one optimization technique we propose is to let different active nodes share as many satellites as possible so that we may have more than k redundant nodes awake when some active node fails, leading to a quicker network recovery. Each satellite will thus have a primary active node, and possibly multiple secondary active nodes. Specifically, if we have n_A active nodes, then the total number of satellites is $n_A k$, but each active node may have more than k satellites. In order to implement this optimization, every satellite should periodically probe to see whether it has more than one active node. If yes, it should register itself with those active nodes, and make them its secondary active nodes.

The second issue we need to address is how to integrate sleeping schedules for redundant nodes. A redundant node can belong to several active nodes at the same time. When a redundant node wakes up, if it can go back to sleep, all the associated active nodes should notify it with corresponding following wake up times. Please note that only the one that expects the redundant node to wake up at this time re-calculates its sleeping time as mentioned above, and all the others just re-send the calculated sleeping intervals.

Finally, another optimization technique we can apply is that, instead of letting the satellites stay active all the time, we can make them sleep as well. On the other hand, however, these nodes should sleep for short time and wake up very often. This way, we can further cut down the energy consumption and lengthen the network lifetime.

4.3 Asynchronous Wakeup Algorithm

Instead of guaranteeing lossless network operations, asynchronous wakeup algorithm takes a different approach: it allows to have periods during which network coverage or/and connectivity is lost, but it guarantees an upper limit on the duration of these periods (denoted by Δ). Specifically, it ensures that, after any active node fails, k (i.e., safety degree) redundant nodes will be awaken within a Δ interval. Therefore, every redundant node is treated equally, and they all go to sleep and wake up periodically.



(a) Wake ups are clustered. (b) Wake ups are uniformly spread out.

Figure 5: Wake ups should be spread out rather than clustered. In this example, active node A has three redundant nodes: B, C, and D. $n = 3$ and $k = 1$, and sleep interval is 3Δ. (a) shows that a clustered schedule cannot provide the upper bound of Δ.

The key parameter for asynchronous wakeup is the sleep interval for each redundant node. Let us consider an example: active node A has n redundant nodes, and we need to make sure that, after A fails, k (out of n) nodes wake up within Δ. In order to have k wake ups within Δ, we should let every node wake up every $\frac{n\Delta}{k}$ interval. Obviously, this method makes those nodes that have more neighbors sleep longer than those that have fewer neighbors, and the resulted spatial diversity can help strike the balance between network operations and network lifetime. Another point we would like to emphasize is that this sleep interval ($\frac{n\Delta}{k}$) is very small, and it is only needed when the active node fails or is about to fail. If the redundant nodes employ such a small interval all the time, then it leads to an excessive number of wake ups, and an enormous amount of energy will be consumed. Instead, we should make redundant nodes wake up less often if the active node is unlikely to fail in the near future. For instance, if A (the active node in our example) just became active, then the chances are it can last for a reasonable amount time, so its redundant nodes do not need to wake up this frequently. In order to capture this behavior, we adopt a threshold-based scheme to adapt the sleep intervals temporally. Our rationale is that, the remaining lifetime of a node is proportional to the remaining energy it has. We use TH_{energy} to denote the threshold for remaining energy. If the remaining energy of the active node is below the threshold, we adopt a smaller sleep interval to guarantee the network operation loss upper limit; otherwise, we adopt a larger sleep interval to conserve energy. More formally put, we can calculate sleep intervals as

$$T = \begin{cases} \frac{n\Delta}{k} & e < Th_{energy} \\ \frac{k_1 n\Delta}{k} & \text{otherwise} \end{cases},$$

where k_1 is a tunable parameter. All three parameters in this algorithm, i.e., Δ, TH_{energy}, and k_1, can be easily tuned by the applications. For instance, those applications that weigh lifetime more than continuous coverage/connecitivy provisioning would choose a large Δ, a small TH_{energy}, and a large k_1.

Although the basic idea is rather simple, we need to address a few issues that are related with the implementation of this algorithm. The first issue is whether some level of coordination between the redundant nodes is needed, or a completely distributed solution is enough. Figure 5 illustrates that while a proper sleep interval is important, how the wake ups are distributed in time is even more important. In both scenarios, the same sleep interval is used (3Δ), but one schedule produces clustered wake ups, and

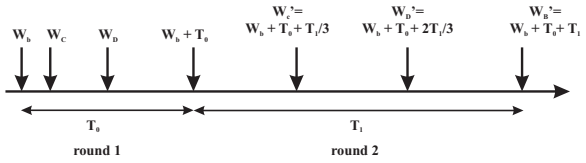


Figure 6: Wake ups are made uniform in the second round.

it fails to deliver the promised coverage/connectivity loss upper limit (Figure 5(a)). Instead, the other schedule produces evenly distributed wake ups, and it meets the requirement (Figure 5(b)). It is hard for the redundant nodes themselves to coordinate their wake ups because they are not aware of each other's schedules. In addition, the set of redundant nodes may change over time because these nodes may fail or become active. Therefore, a completely distributed algorithm is not a viable alternative. Instead, we need to impose centralized coordination between the redundant nodes, and the best candidate to perform this coordination is the corresponding active node. In this light, every active node should address the following questions:

- *How many redundant nodes does it have?*

  Every active node maintains a redundant node table. The table has three fields: *ID* which denotes the redundant node ID, W_{last} which denotes the last wake up time stamp of that redundant node, and W_{next} which denotes the next wake up time of that redundant node. Every time when a redundant node wakes up, it registers itself with the corresponding active node(s). If it is a new member, a new row will be created in the table; otherwise, its time stamps will be updated. If the redundant node's sleep interval is T, then the active node can hear from all of its redundant nodes within T. Consequently, if a redundant node is not heard for a period of cT, the active node assumes it has left and erases it from the table, where c is a tunable parameter which can have a small value.

- *How to uniformly spread out the wake ups?*

  To simply the explanation, we use the example scenario illustrated in Figure 5, i.e., active node A has three redundant nodes B, C and D. Using the same example, our algorithm is illustrated in Figure 6. A employs the concept of "round" to make the wake ups uniformly spread out. From its redundant node table, A chooses the one with the smallest ID (B in our example), and the wake up of B marks the beginning of a new round. At the beginning of round i, A determines B's sleep interval in the NEXT round $i+1$ using the algorithm described above. In Figure 6, time stamp w_B marks the beginning of round 1, and T_1 is determined at the same time. In round 1, however, B's sleep interval is T_0, which was determined in the earlier round. Then B's next wake up is at $w_B + T_0$, and its subsequent wake up is at $w_B + T_0 + T_1$. Our algorithm makes sure that C and D wake up after w_B, but before $w_B + T_0$ in round 1. Without loss of generality, we assume that C wakes up at w_C, D at w_D, and $w_B \leq w_C \leq w_D < w_B + T_0$. When C wakes up, A will record w_C in the field of W_{last} in the redundant node table, and set its next wake up time as $w_B + T_0 + \frac{T_1}{3}$. This way we ensure that C's next wake up is in round 2. As a result, C's next sleep interval is $w_B + T_0 + \frac{T_1}{3} - w_C$. Similarly, D's next wake up time will be set as $w_B + T_0 + \frac{2T_1}{3}$, and its next sleep interval will be $w_B + T_0 + \frac{2T_1}{3} - w_D$. Using this algorithm, the wake up times of B, C, and D are evenly distributed.

  In summary, repeating this simple algorithm can ensure that every node wakes up uniformly within each round.

In the above algorithm, we deal with the situation when the set of redundant nodes does not change. In reality, however, redundant nodes may leave dynamically, and this is the second we need to address. For example, in Figure 6, if C leaves in round 2(before its wake up time), then the gap between B's wake up and D's may be greater than Δ, thus violating the upper limit guarantee. A node may leave the redundant node set for two reasons: (1) it fails; or (2) it becomes active. In the case of (2), we do not need to modify the algorithm because that node will be active during emergency so it can help anyway. In the case of (1), we can tailor the algorithm suitably by taking the remaining energy of redundant nodes into consideration. If a redundant node's remaining energy is below a certain threshold, we can adopt a smaller sleep interval than $\frac{n\Delta}{k}$ to minimize the likelihood of the violation.

The third issue is that a redundant node may belong to more than one active node. As an example, a redundant node B may be associated with two active nodes: A and E. Every time when B wakes up, both A and E will compare the current time stamp (t_{curr}) with the corresponding W_{next} field in its redundant node. Suppose that the W_{next} field in A's table is greater than t_{curr}, then A will not update its table, but simply send $W_{next} - t_{curr}$ as the next sleep interval to B. At the same time, E will find that its W_{next} field is the same as t_{curr}, and it will calculate the next wake up time for B as mentioned above, update the table according, and send the new sleep interval to B. Consequently, B receives two sleep intervals from both active nodes every time it wakes up, and it sleeps according to the smaller value.

5 Simulation Results

We have conducted numerous experiments to validate and evaluate the proposed heuristics. In this section, we report detailed simulation results.

5.1 Node Failure Model

It is a challenging task to smoothly construct a new set of active nodes from the redundant nodes because these redundant nodes may be sleeping when the current active node fails. It is made even more challenging by the fact that failures are no longer an exception with sensor nodes. Sensor nodes are likely to die before they run out of limited battery power ([7, 27, 2]) since the sensor hardware is not reliable and the physical environment in which they are deployed is harsh. In this paper, we use node *failure* to denote that a node dies before its battery is drained out. A good node scheduling algorithm must survive frequent node failures. The characteristics of failure distributions have a significant impact on the network behavior. In other domains, researchers have been using the real failure trace to drive their studies, but such data is not yet made available to us in the sensor

applications. As a result, we use two synthetic failure models in this paper to investigate whether our schemes can survive various failure scenarios.

- **Aging Failure Model**

  Failures on other computing platforms such as clusters of PC boxes have been studied for decades, and it has been shown that a node that has run for a significant period of time is more likely to fail than those that have run for a short time frame [5] due to the wear-out effects on the hardware. Inspired by this observation, we come up with the aging failure model. Suppose that a sensor node starts with an initial energy E, and that E_0 amount of energy still remains when it dies due to hardware reasons. The ratio $x = \frac{E_0}{E}$ thus indicates how soon a node dies before its battery runs out, and we call this the *failure rate* of that node. In this model, x is a random variable uniformly distributed between 0 and 2μ, where μ is the average failure rate. We can adjust the value of μ to obtain different network failure rates.

- **Catastrophic Event Failure Model**

  While the aging failure model captures failures due to the hardware wear-out effect (especially when the hardware is unreliable), the catastrophic event failure model captures failures caused by external events, such as sudden temperature increase, high degree of humidity, lightning, flooding, etc. We propose to use two parameters to model the occurrences of the external catastrophic events. The first parameter is the mean time between catastrophic events, which follows an exponential distribution with mean of XXX seconds. The second parameter is the percentage of the nodes that are affected by a particular event, no matter whether it is awake or sleeping. We use the random variable x to represent this percentage, and x is called the system *failure rate*, uniformly distributed between 0, and 2μ, where μ is the average failure rate.

5.2 Performance Metrics

In this paper, we use the following metrics to evaluate different energy conservation protocols:

- **Network Lifetime**. A sensor network is alive when there exists a routing path from the source to the sink node. One hopes the average network lifetime scales with the number of nodes deployed, and degrades gracefully with the increasing failure rates.

- **Average Event Delivery Ratio**. The average event delivery ratio measures the connectivity maintenance of the network. It is the ratio of the number of distinct events (messages) that are received by the sink to the number of distinct events (messages) that are sent by the source node(s).

- **Average Coverage Recovery**. This metric measures how quickly the redundant nodes can become awake after an active sensing node fails. The smaller the recovery time, the better network coverage is maintained.

5.3 Simulation Environment

We have implemented the two DADA heuristics using the ns-2 simulator.

Sensor Node Configuration. We select sensor node parameters similar to those of Berkeley Motes [1]. Node radio range is 15m (we do not differentiate radio range and interference range in this paper). Sensing range is also 15m. The transmission, reception, idle, and sleep power consumptions are 60mW, 12mW, 12mW, and 0 respectively. Initial energy is randomly chosen from 40J to 60J, but sinks have infinite amount of energy. We employ 1 Mbps IEEE 802.11 as MAC layer protocol. For the catastrophic event failure model (Section 5.1), the mean catastrophic event inter-arrival time is 5000 seconds (which is approximately equal to a sensor node's life time).

Topology. We simulate a $50 \times 50m^2$ field, and the size is fixed across all the simulations.

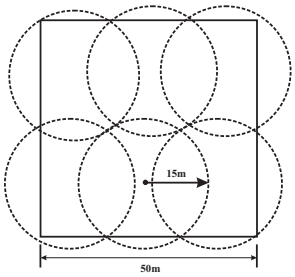


Figure 7: Coverage

Investigating what regions should be covered is beyond the scope of this paper. In order to test the efficiency of DADA in maintaining continuous coverage, we simply partition the sensor field into 6 regions, illustrated by the circles in Figure 7, and we want to make sure there is at least one active sensing node within each circle.

In order to test DADA's efficiency of extending lifetime while maintaining a high connectivity level, we use the following two scenarios: (1) single source and single sink (S-TOPO in short), where the source is randomly chosen from one of the 6 sensing circles; and (2) multiple source and multiple sink (M-TOPO in short), where all the 6 sensing circles are reporting their data to the 3 sinks which are uniformly distributed within the network field. For each scenario, we vary the traffic pattern, number of nodes, and node failure rates. We compare our schemes with ASCENT [2], which employs a fixed sleep interval for all the redundant nodes. We also compare these energy-aware schemes with the scenario where all the deployed nodes are kept active, which is referred to as *BASIC* scheme in this paper.

Routing Protocol. We use Directed Diffusion [8] as our routing algorithm. A data message has 64 bytes, and an interest message has 36 bytes. DADA is placed directly above the MAC layer, and below the routing layer.

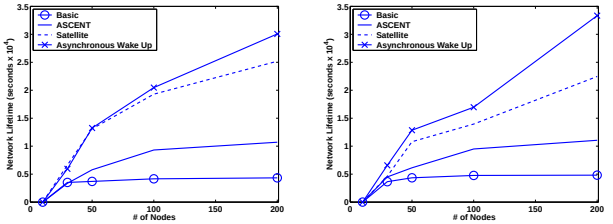
Traffic Pattern. In order to model realistic network settings and demonstrate that our heuristics can be applicable to more generic situations, we use both constant bit rate (CBR) and variable bit rate (VBR) sources. A CBR source sends a packet every 5 sec-

onds. Again, we would like to emphasize that examining the interference between DADA and congestion is part of our future work. The packet inter-arrival time of a VBR source follow an exponential distribution with the mean of 5 seconds. We purposefully choose exponential distribution for its high variance.

Algorithm Parameters. For all four schemes except BASIC, data delivery ratio is calculated every 50 seconds. For ASCENT, the sleep interval is 200 seconds, test period 50 seconds, and passive interval 70 seconds [2]. For asynchronous wake-up, the lifetime threshold ($TH_{lifetime}$) is 350 seconds, and the maximum tolerable connectivity loss period (δ) is 70 seconds.

5.4 Energy Conservation

In this section, we compare the efficiency and scalability of the energy conservation capability of four schemes (BASIC, ASCENT, satellite, and asynchronous wake-up).



(a) Aging - failure rate = 0.15

(b) Catastrophic event - failure rate = 0.15

Figure 8: Efficiency and scalability of energy conservation. S-TOPO topology, CBR source.

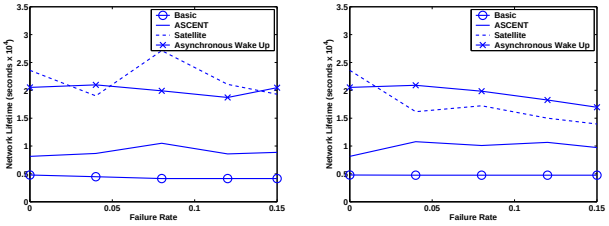
Figures 8(a) and (b) compare the four schemes in terms of energy conservation efficiency and scalability. In this set of experiments, we use S-TOPO topology and CBR sources.

BASIC is not a viable approach. BASIC provides the shortest network lifetime, and its network lifetime does not scale with the redundancy degree because every sensor node is active.

ASCENT improves network lifetime significantly compared to BASIC. However, since it uses a simplistic fixed sleep interval for every sensor node, its improvement cannot scale beyond 100 nodes. Our simulation results confirm the analysis given in [2]: the lifetime improvement factor is limited by the ratio of the sleep interval and the passive interval when these intervals are fixed. In our experiments, sleep and passive intervals are 200 and 70 seconds respectively, which gives the theoretical upper bound of 2.8. The gap between this upper bound and the actual observed value (2.6) comes from the failures.

The two heuristics of DADA, on the other hand, significantly prolong the network lifetime. More importantly, both schemes scale well (close to linear) with redundancy level, especially for asynchronous wake-up. At node count 200, under both failure models, asynchronous wake-up can improve the lifetime by a factor of 7.5 compared to BASIC, and a factor of 2.9 compared to ASCENT.

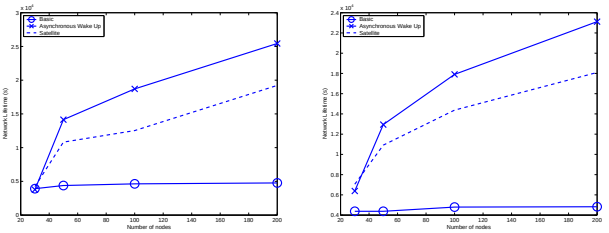
Unreliable hardware and harsh environments have increased the possibility for a sensor node to die before its battery is drained out. Schemes that take these dynamics into consideration can maintain a desirable connectivity and a short network recovery time. The impact of node failure rates on network lifetime has been studied, and the results are provided in Figures 9(a) and (b).



(a) Aging - node count = 100

(b) Catastrophic event - node count = 100

Figure 9: The impact of node failure rates. S-TOPO topology, CBR source.



(a) Catastrophic event - failure rate = 0.15, M-TOPO, CBR

(b) Catastrophic event - failure rate = 0.15, S-TOPO, VBR

Figure 10: Network lifetime with other scenarios.

Both BASIC and ASCENT are expected to have a constant network lifetime regardless of the failure rate. BASIC keeps all the nodes active, so it has a great level of redundancy to tolerate a large number of node failures. ASCENT employs a fixed sleep interval throughout the execution, so that the failure rate does not affect its performance. More importantly, we find that DADA heuristics are very robust as well against the unexpected failure rates, although they employ a more aggressive sleeping method. This is due to the fact that both heuristics guarantee the active nodes are backed up by some awake redundant nodes as discussed in Section 4.

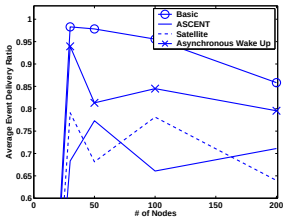
In order to test our heuristics under different network settings, we vary the topology and the traffic pattern. Sample results are provided in Figures 10(a) and (b). The two proposed schemes show similar performance compared to the results shown above (where we use S-TOPO and CBR).

5.5 Connectivity

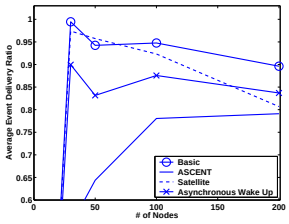
Network connectivity provisioning is measured by the average event delivery ratio. Figures 11(a) and (b) present the results with an increasing deployment redundancy.

As expected, BASIC provides the best delivery ratio among them all because every node is active, except that its average event delivery ratio degrades significantly as the node number increases due to network resource contention. On the other hand, ASCENT has the worst connectivity due to the fixed sleep intervals.

We have showed that DADA heuristics can significantly improve the network lifetime, and we are glad to point out that such



(a) Aging - failure rate = 0.15

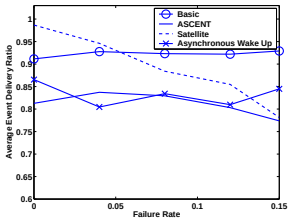


(b) Catastrophic event - failure rate = 0.15

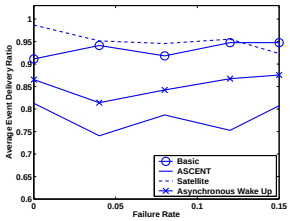
Figure 11: The impact of redundancy level on network connectivity. S-TOPO topology, CBR sources.

significant gain in network lifetime is achieved with a marginal degradation of the event delivery ratio compared to BASIC. Asynchronous wake-up can provide more than 85% delivery ratio with catastrophic failure model, and above 80% under aging model. With catastrophic failure model, the delivery ratio of Satellite scheme is above 90% at node count 100.

Finally, we would like to point out that failure models have different impacts on the performance. For instance, Satellite scheme works better with catastrophic event model than with aging model in terms of event delivery ratio because the latter makes the active nodes and satellites more susceptible to failures. (This can be addressed by increasing the number of satellites an active node can have.)

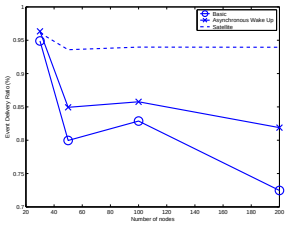


(a) Aging - node count = 100

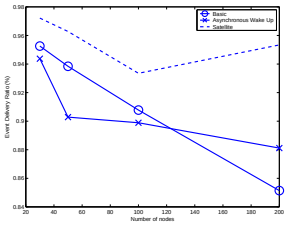


(b) Catastrophic event - node count = 100

Figure 12: The impact of node failure rates on network connectivity. S-TOPO topology, CBR source.



(a) Catastrophic event - failure rate = 0.15, M-TOPO, CBR



(b) Catastrophic event - failure rate = 0.15, S-TOPO, VBR

Figure 13: Network connectivity with other scenarios.

Figure 12 presents the network connectivity when the failure rates increase. For the reasons we have mentioned in Section 5.4, event delivery ratios of BASIC and ASCENT will not change when node failure rates increase. Asynchronous wake up also can maintain good connectivity even under high failure rates. The only situation where we observe a degradation is when satellite heuristic is used with aging failure model. When aging failure model is used, active nodes and satellites have a higher chance to fail since they have a higher load. However, this can be addressed by increasing the number of satellites. In addition, we would like to point out that even under degradation, satellite provides a better connectivity than ASCENT.

A similar trend is observed when we use different network topology (M-TOPO) and traffic pattern (VBR), as shown in Figure 13.

5.6 Coverage

We have conducted experiments to study the efficiency of satellite and asynchronous wake up in providing continuous network coverage. We are not intending to define the set of active nodes that can cover the entire sensor field; instead, once these nodes are identified we can guarantee the same spots will be continuously monitored even after the initial set of active nodes die. The design of both satellite and asynchronous wake up ensures that multiple redundant nodes are awake when node failures occur. We have looked at the gap between active sensing node failure and redundant nodes wake up, which is referred to as *coverage recovery time* and the average coverage recovery time is negligible with satellite heuristic since the satellite nodes wake up very frequently. The average coverage recovery time with asynchronous wake up is always less than δ.

6 Related Work

Energy vs. Coverage Energy-efficient approaches to maintaining network coverage have been extensively studied [7, 3, 19, 27]. LEACH [7] forms clusters and rotates cluster leaders to balance the energy consumption between cluster members. SPAN [3] turns off those nodes that do not improve coverage. In [19], Tian proposes to select redundant nodes by comparing each other's sensing range, and then to turn off the redundant nodes to conserve energy. In order to avoid blind spots, they use a backoff based scheme.

PEAS [27] assumes that every node has already obtained the knowledge about who are redundant to itself. PEAS attempts to ensure only one node stays active from any sensing range. Redundant nodes thus alternate between sleeping and wake up. After a redundant node wakes up, it sends out a probing message asking if there is any active node. If it receives the response, then it will go back to sleep. Otherwise, it will become active. It uses reliable protocols to ensure that these probing messages will not get lost.

In [26], a round-robin like schedule is proposed among all the redundant nodes so that every node knows when it should become active and how long it should stay active. Such a fixed schedule is efficient for scenarios with low network dynamics.

Energy vs. Connectivity A number of energy-aware connectivity provisioning techniques have been proposed. GAF [25] proposes to identify the nodes that are equivalent in terms of routing functionality, and then turn them off to conserve energy without interfering with connectivity. In AFECA [24], a simple

adaptive scheme is used to determine the node sleep interval. Every redundant node counts how many neighbors it has and then adapts its sleep interval based on that count. A redundant node can count its active neighbors by listening to the packet delivery after it wakes up.

ASCENT [2] shares the same goal with GAF, and it also considers the mechanisms for the sleeping nodes to turn back on and join the routing when necessary. These nodes will wake up periodically. When they are awake, they test the local link quality by observing the packet delivery ratio; if the observed packet delivery ratio is below a certain threshold and the number of active neighbors is low, the node will become active. In order not to degrade connectivity significantly, sleeping nodes need to wake up relatively frequently. In addition, a fixed sleep interval is used for every node in the network. The authors also proved that the upper bound on the lifetime improvement is the ratio between a node's sleep interval and its awake period.

Coverage and Connectivity A considerable amount of research effort has been devoted to investigating the synergy between network coverage and connectivity. In [22], Wu proposed an approach to calculate the Connected Dominating Set (CDS) for ad-hoc networks. The property of CDS is that all the nodes within the network are either in this set or are the neighbors of the nodes that belong to the set. CDS can be used to design efficient routing protocols. Built upon CDS, [10] proposes a novel way of forming hierarchies in sensor networks based on the roles of each sensor node. [4] gives a mathematical analysis on the relationship between redundancy degree and the number of active neighbors. Considering non-linear energy depletion and unexpected failures of sensor nodes, [4] could serve as guidance to the design of energy conservation heuristics.

In [20], a Coverage Configuration Protocol (CCP) is introduced to provide different degrees of coverage as per the applications' request. A geometrical analysis between coverage and connectivity is provided as well.

7 Concluding Remarks

Compared to previous work, we acknowledge the significance of preserving coverage and connectivity, in the pursuit of extending network lifetime. With proper network coverage, the sensor network is capable of sensing the areas of the network that are of interest to the application. Connectivity will ensure that the desired data will be received as a constant, uninterrupted stream. If we do not considering these factors, trying to improve network lifetime can result in disappointing performance from an application's point of view.

In this paper, we propose a novel 2-dimensional adaptive technique, DADA, that can dynamically calculate a node's sleep interval based on application demands and local network conditions. Using this technique, redundant nodes can sleep longer when the situation allows and wake up more often when it is going to be needed. As a result, the network lifetime can be extended while providing improved coverage and connectivity guarantee. In order to realize the goal of DADA, we propose two heuristics. One heuristic partitions redundant nodes into small groups, and group members can back up each other so that nodes that belong to other groups can sleep much longer. The other heuristic requires every redundant node to make independent decisions.

We evaluate our schemes using various network topologies and traffic scenarios. From our results, we see that our schemes can simultaneously improve network lifetime, while maintaining coverage and connectivity. The lifetime improvement exhibits a near-linear increase when the deployment redundancy increases and this improvement can survive high node failures. When compared to ASCENT, our schemes can extend the lifetime by a factor of 2.9. In addition, the two schemes can maintain better network connectivity compared to ASCENT.

References

[1] TinyOS. `http://webs.cs.berkeley.edu/tos/`.

[2] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring Sensor Networks Topologies. In *Proceedings of IEEE INFOCOM'02*, June 2002.

[3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, July 2001.

[4] Y. Gao, K. Wu, and F. Li. Analysis on the redundancy of wireless sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 108 – 114, September 2003.

[5] S. Garg, Y. Huang, C. Kintala, and K. S. Trivedi. Minimizing Completion Time of a Program by Checkpointing and Rejuvenation. In *Proceedings of the ACM SIGMETRICS 1996 Conference on Measurement and Modeling of Computer Systems*, pages 252–261, May 1996.

[6] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. An Energy-Efficient Surveillance System Using Wireless Sensor Networks. In *Proceedings of the ACM MobiSys 2004*, 2004.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd Hawaii International Conference on System Science*, 2000.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM)*, August 2000.

[9] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-Efficient Comiputing for Wildlife Tracking: Design and Tradeoffs and Early Experiences with Zebranet. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 96–107, 2002.

[10] M. Kochhal, L. Schwiebert, and S. Gupta. Role-based hierarchical self organization for wireless ad hoc sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 7–14, September 2003.

[11] X. Y. Li, P. J. Wan, and O. Frieder. Coverage in Wireless Ad-Hoc Sensor Networks. *IEEE Transactions on Computers*, 52(6), June 2003.

[12] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks. In *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, September 2002.

[13] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler. Wireless Sensor Networks for Habitat Monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[14] A. Salhieh and L. Schwiebert. Power-aware Metrics for Wireless Sensor Networks. *International Journal of Computers and Applications*, 26(4), 2003.

[15] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *proceedings of the ACM MobiHoc Conference*, 2003.

[16] A. Savvides, S. Park, and M. B. Srivastava. On Modeling Networks of Wireless Micro Sensors. In *Proceedings of the ACM SIGMETRICS 2001 Conference on Measurement and Modeling of Computer Systems*, June 2001. short paper.

[17] S. Shakkottai, R. Srikant, and N. B. Shroff. Unreliable Sensor Grids: Coverage, Connectivity and Diameter. In *Proceedings of IEEE Infocom*, April 2003.

[18] S. Singh and C. S. Raghavendra. PAMAS - Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. *ACM SIGCOMM Computer Communication Review*, 28(3):5–26, July 1998.

[19] D. Tian and N. D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 32–41, September 2002.

[20] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks . In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 28–39, November 2003.

[21] A. Woo and D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, July 2001.

[22] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 7–14, August 1999.

[23] G. Xing, C. Lu, R. Pless, and J. A. O'Sullivan. Co-Grid: an efficient coverage maintenance protocol for distributed sensor networks. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 414–423, April 2004.

[24] Y. Xu, J. Heidemann, and D. Estrin. Adaptive energy-conserving routing for multihop ad hoc networks. Research Report 527, USC/Information Sciences Institute, October 2000.

[25] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed Energy Conservation for Ad Hoc Routing. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, July 2001.

[26] T. Yan, T. He, and J. A. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *Proceedings of the ACM SenSys 2003*, 2003.

[27] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, May 2003.

[28] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of IEEE INFOCOM'02*, June 2002.